ECT | Revista Cientifica Emprendimiento Científico Tecnologico

ISSN: 2810 – 8493
https://revista.ectperu.org.pe/index.php/ect/index

A Card Game Proposal for Approaching Regular Languages

1st Valentina Narvaez-Teran
Computer Sciences Department
Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias
Monterrey, Mexico
valentina.narvaez@tec.mx
https://orcid.org/0000-0003-2071-9568

2nd Lorena Martinez Elizalde
Computer Sciences Department
Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias
Monterrey, Mexico
lorenamtze@tec.mx
https://orcid.org/0009-0006-4024-4271

***Abstract*—Formal languages are a relevant subject in computer science programs, they are a fundamental cornerstone for the development of programming languages, pattern recognition in texts, and they are deeply related to the theory of computation, including problems and their complexity. This work proposes a turn-based competitive card game that models the basics of regular languages, also known as type-3 languages. Through this game's mechanics, we aim to model relevant formal concepts, such as languages, symbols, alphabets, words, operators and word matching with regular expressions. We present the rules of the game and the design reasoning behind them, explaining how this rules relate to the abilities students require to work with regular languages. We report some early feed back received from computer science students.**

***Index Terms*—Gamification, educational game design, regular languages**

I. Introduction

Gamification in education has been proposed as a strategy to motivate students, increase engagement and facilitate the understanding of abstract concepts [1]. Translating such concepts into game mechanics can help students approach a subject that they may perceive as too complex or too unfamiliar from a new perspective [2]. In today's college classrooms, where students may struggle to overcome learning issues [3], such as decreasing attention spans, easy access to distracting devices, difficulty to connect with traditional learning methodologies, and plain old lack of interest, games can be a valuable alternative tool for professors. Learning via games can also help remove communication barriers, such as student's fear of being judged if they admit they do not fully understand a subject [4].

This work presents preliminary work on the development of a card game aiming to teach the operators involved in regular languages, and the regular expressions that describe them. These subjects are a relevant part of college programs for computer sciences majors and other related disciplines. While the game was developed in the context of higher education, we consider it is simple enough for younger players, and requires no prior knowledge.

Regular languages are the starting point to approach Chomsky's classic hierarchy [5]–[7], as its least complex member.

ECT | Revista Cientifica Emprendimiento Científico Tecnologico
ISSN: 2810 – 8493
https://revista.ectperu.org.pe/index.php/ect/index

A good grasp of formal languages is vital for pivoting later to related subjects, such as the development of programming languages [8] and automata theory [9]. Therefore, our game can be potentially useful in computing theory and programming languages courses. Regular languages include sequences of symbols matching certain patterns. This symbols must belong to a given alphabet, and the patterns are based on simple operations over them: concatenation, repetition and alternative.

The rest of this paper is organized as follows. Section II introduces regular languages and their terminology. Section III describes the rules involved in the game and how they model the basic concepts of regular languages. Student's initial feedback and preliminary results are discussed in section IV. Finally, section V summarizes conclusions and further work.

II. Formal definitions and operators

This section defines the concepts related to regular languages that we attempted to capture in the game's mechanics.

- **Alphabet**. An alphabet Σ is a finite non-empty set of symbols.
- **Words**. A word $\omega \in \Sigma$ is a sequence of symbols from the alphabet, and its length is denoted as $|\omega|$.
- **Empty word**. The empty word ϵ has length zero.
- **Power operator**. The power operator can be applied to any alphabet to describe words of specific length. For ex- ample, Σ^3 is the set of all words of length equal to three, formed with symbols from Σ. This operator can also be applied to symbols, either single or grouped within parenthesis. For example: a^5 results in five repetitions of the a symbol, i.e., *aaaaa*. Meanwhile, $(ab)^2$ would result in two repetitions of the *ab* group, i.e., *abab*.
- **Kleene's closure**. Also known as Kleene star. It is the union $\Sigma* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdot \cdot \cdot \cup \Sigma^n$, with $0 \leq n$. It represents all possible words of any length over the symbols of alphabet Σ. Notice $\Sigma*$ includes the empty word ϵ.
- **Positive closure**. Also called Kleene plus. It is the union $\Sigma* = \Sigma^1 \cup \Sigma^2 \cup \cdot \cdot \cdot \cup \Sigma^n$, with $1 \leq n$. It represents all possible words of length equal or larger than one, over

ECT | Revista Cientifica Emprendimiento Científico Tecnologico
ISSN: 2810 – 8493
https://revista.ectperu.org.pe/index.php/ect/index

the symbols of alphabet Σ. Since $1 \leq n$, Σ+ does not include the empty word ϵ.

- **Concatenation**. It is the sequential combination of several symbols and/or words. For example ab is the concatenation of the symbol a followed by the symbol b. The concatenation of a word ω with the empty word ϵ is the word itself: $\omega\epsilon = \omega$.
- **Alternative**. It is denoted by the | (pipe) symbol. For example, $b \mid c$ describes one occurrence of b or c (but no both together). A patter such as $a(b \mid c)$ matches the words ab and ac.
- **Language**: A language is defined as $L \subseteq \Sigma^*$, where Σ*, also known as Kleene's closure, represents all possible sequences of any length over the symbols of alphabet Σ.
- **Regular languages and regular expressions**. A regular expression is a description of the pattern followed by the words of a regular language. It recognizes all words matching the pattern and it rejects all others. Regular expressions are built from symbols of the alphabet combined with operators, and parenthesis to group them if required. For example, the expression $a*b^5 \mid c^+$ describes all words that:
  - begin with zero or more occurrences of a, followed by either
  - five occurrences of b, or
  - one or more occurrences of c.

  Some of these words are $abbbbb$, ac, $accc$, $aaaaac$, and simply c. The later is the shorter word that matches the expression.
  The patterns defined by regular expressions are relatively simple, but powerful enough to represent the lexicon of programming languages, such as the conventions for naming variables, different types of numbers (integers, floats, with or without sign), etc. They are also useful to validate if user input fits a desired pattern: a valid email, or a strong password.

III. The card game and its rules

The game aims to model the concepts and operations described in Section II, (excluding parenthesis) and to trigger the pattern recognition process required to verify matches. However, it does not employ characters as symbols, nor the explicit notion of expression. Instead, it focuses on capturing cats by satisfying their meal requests considering several types of food, quantity and order. Food types represent alphabet symbols, with meal requests as a metaphor for regular expressions. The game was created for four players. Its development is still at the prototype stage, but it is already playable.

A. Set of cards and tokens

A full set of cards includes the four subsets described below.

- **Cats.** There are ten cat cards. A cat is worth a set amount of points, specified in the left corner of the card.
- **Food tokens.** 24 cards. There are four types of food: fish (f), chicken (c), milk (m) and dry feed (d), with eighth copies per type. Together with the operator tokens, these are used to generate randomized meal requests.
- **Operator tokens.** 13 cards: four copies of the Kleene star operator ($*$), four copies of the Kleene plus operator ($+$), three copies of the alternative operator ($|$), 3 copies for power two (2) and 2 tokens for power three (3).
- **Food cards.** 80 cards, with 20 copies per food type. This deck is used by the players to create matching sequences for a cat's meal request.

Fig. 1. Types of cards: cats, food, food tokens (small and square) and operator tokens.

In order to quickly develop an initial prototype, all the images for the cards were produced using generative artificial intelligence via Microsoft's Copilot.

B. Game objective and rules

The game's objective is to accumulate the most points by gathering as many cats as possible. Cats are won over if their meal request is satisfied by matching the type of food, order and quantity. A game lasts ten rounds, one per cat. Players start by separately shuffling every subset of cards and tokens.

- **Cats and the meal requests.** A round begins by drafting and revealing a cat. The cat and its meal request are placed at the center of the table, for all players to see. The meal request is random, chosen by drafting 4 food tokens and 4 operator tokens. The chosen tokens are placed facing down, in alternate order beginning with a food token, then an operator token. After placing them, they are revealed and grouped into "courses". Since there are no parenthesis in the card set, operators only affect the food token immediately before. The only exception to this is the alternative operator.
  Figure 2 shows an example with three courses. The alternative operator in between the first and second food tokens affects both. The meal request is interpreted as: one serving of chicken or one or more servings of dry feed for the first course, followed by two servings of milk for the second course, and finally zero or more servings of fish for the last course. The underlying language is described by the regular expression $c \mid d^+m^2f^*$.

ECT | Revista Cientifica Emprendimiento Científico Tecnologico
ISSN: 2810 – 8493
https://revista.ectperu.org.pe/index.php/ect/index

It is possible that an alternative operator appears at the end of a randomly generated request, with no food token after it. In that case, the absent food token represents the empty word ϵ.

- **Card dealing** Every player gets six cards from the food deck. Players must not reveal their hands.
- **Plating phase** In turns, players add food from their hand to the area in the table in front of themselves (their "plate"). Cards added to the plate never go back to a player's hand. In order to add food to the plate, the card must match the type of food, one course at a time, while the quantity can be achieved incrementally along several turns. Following the example in Figure 2, a player must begging by plating chicken or dry feed, but no both. A player can not begin by plating milk nor fish, because these do not match the first course. A player can not plate chicken and milk at the same time, since these belong to different courses. If a player does not have the cards required, they must pass the turn.
  The plating mechanic tries to model how regular languages function. Sequences of symbols must follow patterns, where the order, combination and number of repetitions matters. In the previous example a matching word for expression $c\,|\,d^{+}m^{2}f*$ must begin with either one c, or at least one d. Therefore, a matching word can not begin with just mm.
- **Stealing phase** After every player had one turn in plating phase, they move on to stealing phase. In their turn, a player can choose to sacrifice one pair of any type from their hand to steal one card from another player's plate. Sacrificed cards are discarded. The stolen card must be a valid immediate addition to the player's plate. For example, lets say player-A placed one serving of chicken in the plating phase. They can not steal another serving of chicken from another player, since the request only asked for one serving, having two does not match. Player-A can not steal dry food either, because they already choose the chicken alternative.
  A player can steal up to three times in their turn. Stealing repeatedly from the same player is allowed. Stolen cards can belong to different courses. The purpose of stealing is to encourage students to think about how make an existing sequence longer by adding valid symbols.
- **Next phase and turn rotation** After stealing phase, a new plating phase begins. The first turn goes to the player at the left of the one who started the last time. Players add cards to their current hands until completing six.
- **Destroyed plates** A player's plate can be left empty, or with an invalid sequence after the stealing phase ends. If the plate is empty, during the next plating phase the player must start from the first course. If the player was left with something invalid, lets say only a serving of milk, they must focus on rebuilding, starting with the first course, then the next one. If the player has cards for chicken and milk in their hand, they can plate them at the same time. But they can not plate just milk, nor fish.

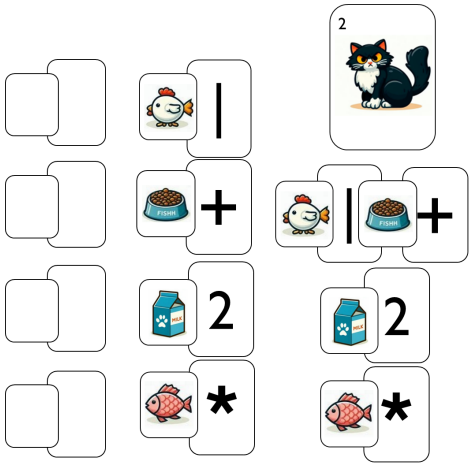

Fig. 2. Steps for drafting and revealing a meal request.

- **Full plate** When a player finishes a plating that fully satisfies the request, they have the chance to win over the current cat. They must announce the plate is ready, in their turn (either in plating or stealing phase). Otherwise, they miss the chance until their next turn.
- **Contested plates and ties** After a player announced they have a ready plate, other players can contest it, by achieving a fuller ready plate before the end of the stealing phase in the current iteration. A fuller plate means a longer sequence of symbols, i.e., a longer matching word. When a player announces their plate is ready, they loose their turn for the stealing phase and the plate is protected, preventing other players from stealing from it. At the end of the stealing phase, the player with the fullest ready plate wins over the cat. However, if players are tied, the game continues as usual for a new iteration of plating and stealing. Tied players recover the ability to stealing and being stolen from.
- **Winning** Players are encouraged to verify if announced ready plates actually match the meal request. If all players agree, the current cat is awarded and the game continues with a new cat and meal request after reshuffling cards.

IV. Early testing and student feedback

A prototype was tested with fourth semester college students who were already aware of regular languages, operators and word verification. While more testing is definitely required, we present here some preliminary results and feedback from 27 students who voluntarily answered a poll. The poll asked students if they agree with the following statements, with answers in ascending scale of 1 to 5. Statements 1 to 5 are about the game and its rules, statements 6 to 11 inquire about student's ability to understand the operators and work with expressions.

1) The game was fun.
2) Rules were easy to understand.

ECT | Revista Cientifica Emprendimiento Científico Tecnologico
ISSN: 2810 – 8493
https://revista.ectperu.org.pe/index.php/ect/index

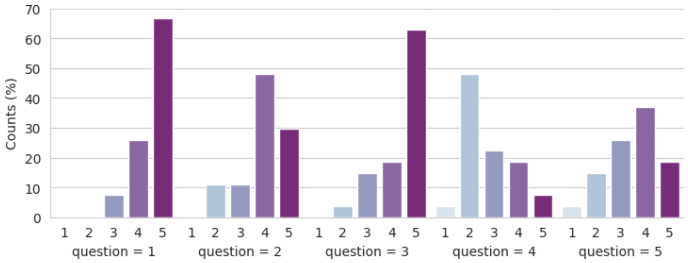

Fig. 3. Percentage of students answers about game mechanics.

3) Food types variety is adequate.
4) It was hard to get a hand with useful cards.
5) It was too easy to get pairs for stealing.
6) I am able to interpret Kleene star and Kleene plus.
7) I am able to interpret the power operator.
8) I am able to interpret the alternative operator.
9) I am able to able to verify a given sequence.
10) I am able to able to create matching sequence.
11) I am able to able to create expressions.

The answers were self-reported perceptions, and we have not yet evaluated the potential impact on academic performance after experiencing the game, but student feedback was overall positive. Most students considered the game was fun, ranking it 4 or above. Most of them also agreed that the rules and types of food cards were adequate. Getting a good hand with useful cards was not considered too hard, while getting pairs to use when stealing was considered a little too easy.

Over half of the students self-reported they understood the operators, specially the power operator, as seen in figure 4. They also ranked high their self-perceived ability to verify given sequences and expressions. This can be observed in figure 5 The accuracy of this self-reported abilities has yet to be evaluated.

Anecdotal observations. While playing the game, students became strategic, specially during the stealing phase. They quickly identified the rival player closer to winning, and they were able to form fast evolving alliances to steal exclusively from them. Students argued among themselves, explaining to each other why certain plating choices were or were not valid. Moreover, some students who rarely participate in class became more vocal, asking questions to their peers as well as to the professor. We consider this last point is particularly valuable, easing communication and allowing professors to identify issues students may have to grasp specific concepts.

V. Conclusions and Further Work

This worked presented an early prototype of a card game aiming to model regular languages concepts in the context of computer sciences. While so far student feedback has been positive, the impact of the game on academic performance has yet to be quantified. There are also several areas of improvement for the game. The most important one is adding mechanics for the inclusion of parenthesis, since subsequences

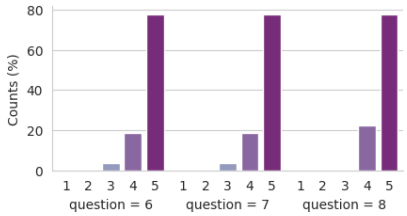

Fig. 4. Percentage of students answers about operators.

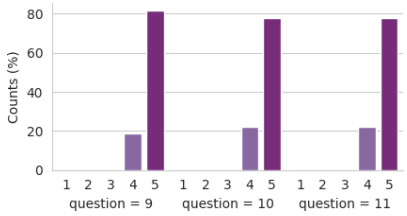

Fig. 5. Percentage of students answers about operators.

of symbols grouped within parenthesis are a vital part of regular expressions. It would be interesting to consider alternatives to add and remove parenthesis in a meal request. More testing with familiarized students and varying group size is also required to ensure the game is adequately balanced. Finally, other objectives are replacing the AI generated images by custom made art designs, making the game publicly available via download, and exploring the possibility of expanding it to cover more complex types of languages.

VI. Acknowledgment

We would like to thank the REDINDVJ for the publication opportunity, as well as the Computer Science Department at Tec de Monterrey for allowing us to try innovations in education through gamification.

References

[1] A. Manzano-Leo´n, P. Camacho-Lazarraga, M. A. Guerrero, L. Guerrero-Puerta, J. M. Aguilar-Parra, R. Trigueros, and A. Alias, "Between level up and game over: A systematic literature review of gamification in education," *Sustainability*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/4/2247

[2] C. Kazimoglu, "Enhancing confidence in using computational thinking skills via playing a serious game: A case study to increase motivation in learning computer programming," *IEEE Access*, vol. 8, pp. 221 831–221 851, 2020.

[3] J. Sinclair, M. Butler, M. Morgan, and S. Kalvala, "Measures of student engagement in computer science," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 242–247. [Online]. Available: https://doi.org/10.1145/2729094.2742586

[4] A. I. Wang and R. Tahir, "The effect of using kahoot! for learning – a literature review," *Computers Education*, vol. 149, p. 103818, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131520300208

[5] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[6] ——, "On certain formal properties of grammars," *Information and Control*, vol. 2, no. 2, pp. 137–167, 1959.

[7] A. V. Aho and J. D. Ullman, "The theory of languages," *Mathematical systems theory*, vol. 2, pp. 97–125, 1968.

ISSN: 2810 – 8493
https://revista.ectperu.org.pe/index.php/ect/index

[8] M. Gabbrielli and S. Martini, *How to Describe a Programming Language*. London: Springer London, 2010, pp. 27–55.
[9] ——, *Abstract Machines*. London: Springer London, 2010, pp. 1–25.